Contextual Knowledge Digitisation and its Application in Software Development

Mr. Manoj Kumar Lal
Author of 'Knowledge Driven Development – Bridging Waterfall and Agile Methodologies'
Consultant, Tata Consultancy Services Limited
manojkumar.lal1@gmail.com

Background

After experimenting with digitisation of contextual knowledge for nearly ten years, I have conceptualised a new proposition around knowledge management named as 'Generic Knowledge Management Framework (GKMF)' and its application in software development named as 'Knowledge Driven Development (KDD)'. Both GKMF and KDD are detailed in my book 'Knowledge Driven Development – Bridging Waterfall and Agile Methodologies' published in June 2018 jointly by IISc Press and Cambridge University Press. I have summarised these concepts again via this article.

Generic Knowledge Management Framework (GKMF)

Generic Knowledge Management Framework (GKMF) is a new framework aimed to capture fit-for-purpose knowledge for a defined scope, generally termed as contextual knowledge. From a structural perspective, contextual knowledge has four layers and eight building blocks as shown in Figure 1 and explained below.

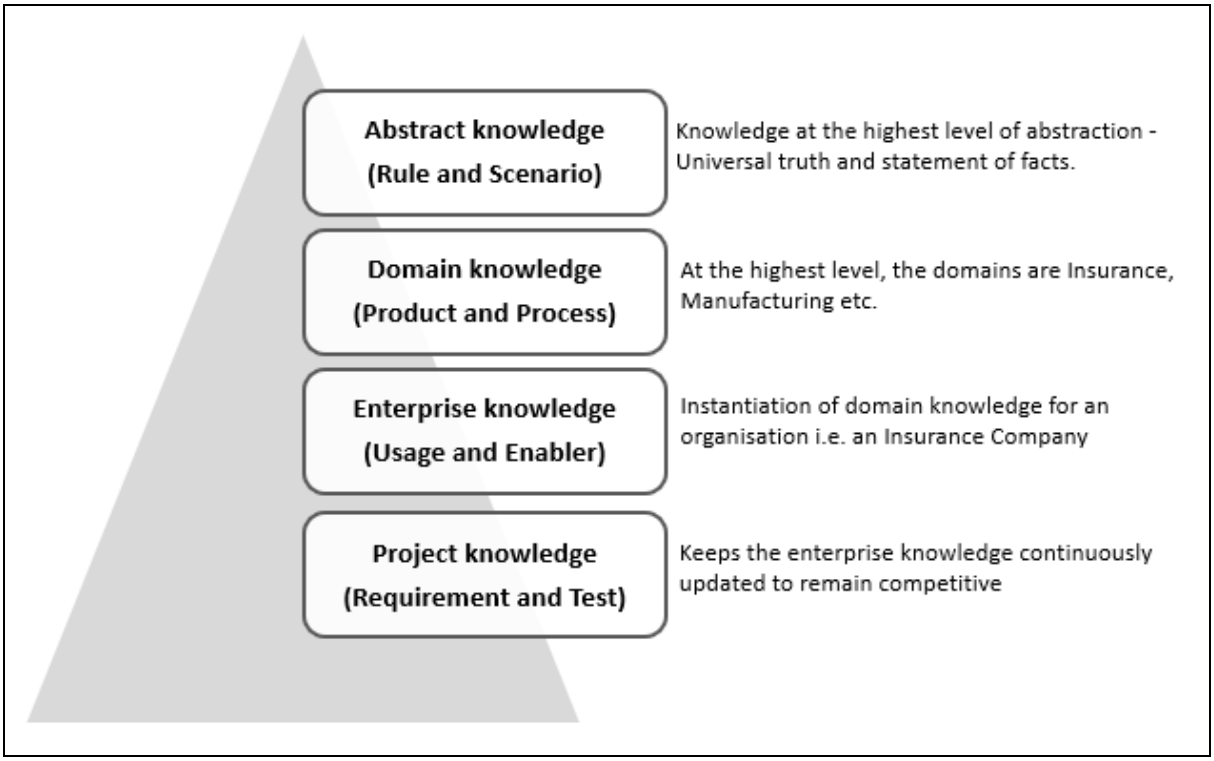

Figure 1: Four layers of contextual knowledge

- Abstract knowledge: Rules and scenarios are enough to capture knowledge at the highest level, i.e., abstract knowledge. They are mostly universal truths and statements of fact. An example can be – Modes of payment via cash, cheque and electronic channel are payment scenario and online shopping accepting only electronic payment is a payment rule.

- Domain knowledge: The addition of products and processes contextualises the rules and scenarios which capture knowledge at a lower level, known as domain knowledge. Knowledge about insurance, banking and manufacturing are examples of domain knowledge. Payment scenario and rule can be contextualised to premium payment process of a life insurance product for insurance domain.

- Enterprise knowledge: Enterprise knowledge is instantiation of domain knowledge for an organisation. The 'Usage' building block provides a mechanism to customise domain knowledge for the organisation via user interface, reports

and communications and 'Enabler' building block automates 'Usage' primarily via IT applications. Knowledge of an insurance company primarily used for its business operations, for example, represents its enterprise knowledge. An example of enterprise knowledge can be – if the insurance premium is more than a certain amount, it must be paid through electronic channel.

- Project knowledge: At the last level comes the project knowledge. The objective of project knowledge is to keep the enterprise knowledge updated to maintain the competitive advantage of the organisation. Project knowledge is driven by specific requirements and a way to prove that the requirements are met is by the 'project test case' building block. An initiative to increase the self-servicing capability of the customer through a customer portal is an example of project knowledge as it reduces the operating cost, assisting in maintaining the competitive advantage of the organisation. Another example of project knowledge can be to create a mobile app for premium payment.

Knowledge levels depend on situation. The project knowledge in one situation can be viewed as abstract knowledge in another. Forcing users to change the password at regular interval – may be project knowledge in the context of an enterprise initiative to enhance security for online users. The same sentence may represent abstract knowledge for a specific project to force password change for the users of a specific IT application. For the same project, an example of project knowledge can be – Users must change password every six months and warning message for the same must start appearing in the last 15 days of reaching six months.

The number of building blocks required to capture a knowledge level will be cumulative blocks of that level and the preceding levels. For example, to capture project knowledge completely, it requires a total of eight building blocks, two from the project knowledge building blocks and the remaining six from the previous knowledge levels. As the project knowledge represents the superset of the building blocks, it is also used to denote the contextual knowledge at some points in this article. As we go down in the level, the granularity of information in the building blocks increases. For example, business rules and scenarios in project knowledge will be more detailed than that of an abstract knowledge.

Execution activities aimed at implementing the contextual knowledge, produce the desired outcome either tangible such as building and factory or intangible such as software and launch of a new brand for an organisation.

Sample applications of GKMF

GKMF has universal applicability. For a defined scope, GKMF aims to capture the contextual knowledge. Some of the examples are detailed below:

A. Let's start with the widest scope and understand GKMF in human lifecycle as shown in Figure 2.

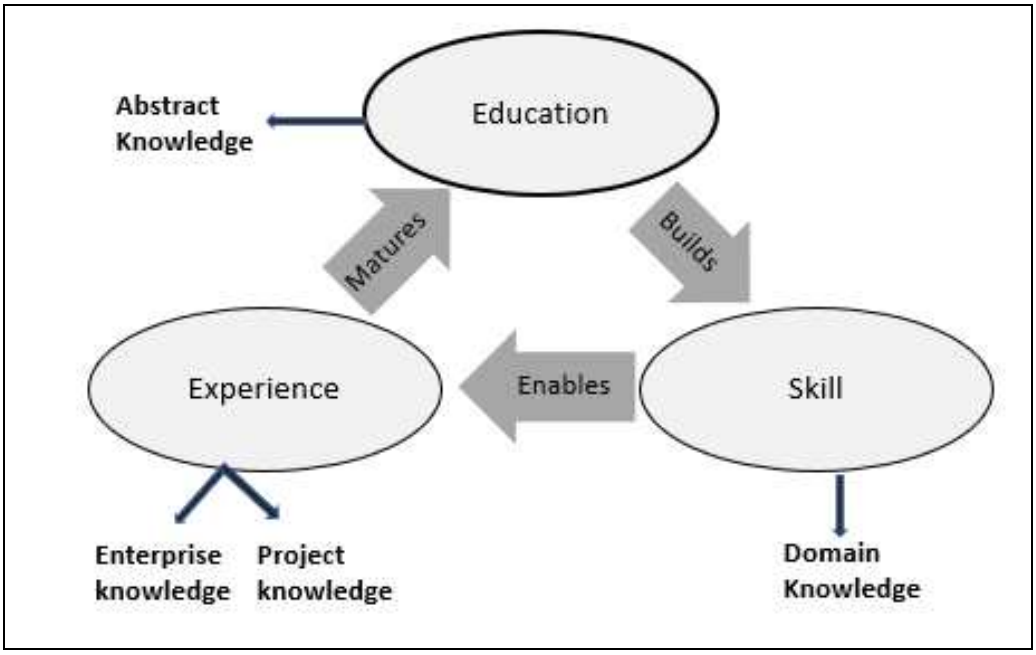

Figure 2: Levels of knowledge in human lifecycle

Knowledge remains relevant to us in various forms - right from our childhood till the end of our active life. In the early stages we gain knowledge via primary and secondary education (abstract knowledge). When we reach adulthood, we learn necessary skills (domain knowledge) to prepare ourselves for the profession of our choice. When we join an organisation, we accomplish tasks and start gaining experience in the skill acquired. There are primarily two types of tasks – repeatable or specialised tasks (enterprise knowledge) and tasks to optimise repeatable or specialised tasks (project knowledge).

B. Limiting the scope to managing details of the customers of an enterprise, the four layers of knowledge are:

- Abstract knowledge**:** Customer details can be of two types – personal details such as date of birth and contact details such as mobile number.
- Domain knowledge: For customers of an insurance company, personal habits such as smoking and having dangerous hobbies influence the premium calculation.
- Enterprise knowledge: Usage of channels in a specific organisation for change in personal details may be - Call centre – 50%, Sending letter – 25%, Doing it online via self-servicing portal – 25%.
- Project knowledge: The company may initiate a project tasked with increasing the online channel usage for change in personal details from 25% to at least 75% to cut expenditure and remain competitive.

C. Narrowing the scope little further to password management, the four layers of knowledge are:

- Abstract knowledge: The password should be strong enough so that it cannot be easily deciphered.
- Domain knowledge: For specialised high cost insurance products a second level password is required.
- Enterprise knowledge: A company has decided that all its Passwords must have 8 characters consisting of a combination of alphabets, numeric and special characters, and should lock after 3 unsuccessful attempts.
- Project knowledge: There is a requirement to send an email to the user immediately after he/she changes their password.

It is clear from the above examples that GKMF helps in scientific evolution (detailing) of the contextual knowledge.

Application of GKMF in software development – Knowledge Driven Development (KDD)

Software is built through a set of execution activities such as build and test based on its contextual knowledge. In Figure 3, the contextual knowledge of GKMF is customised for software development by expanding the building blocks from 8 to 18.

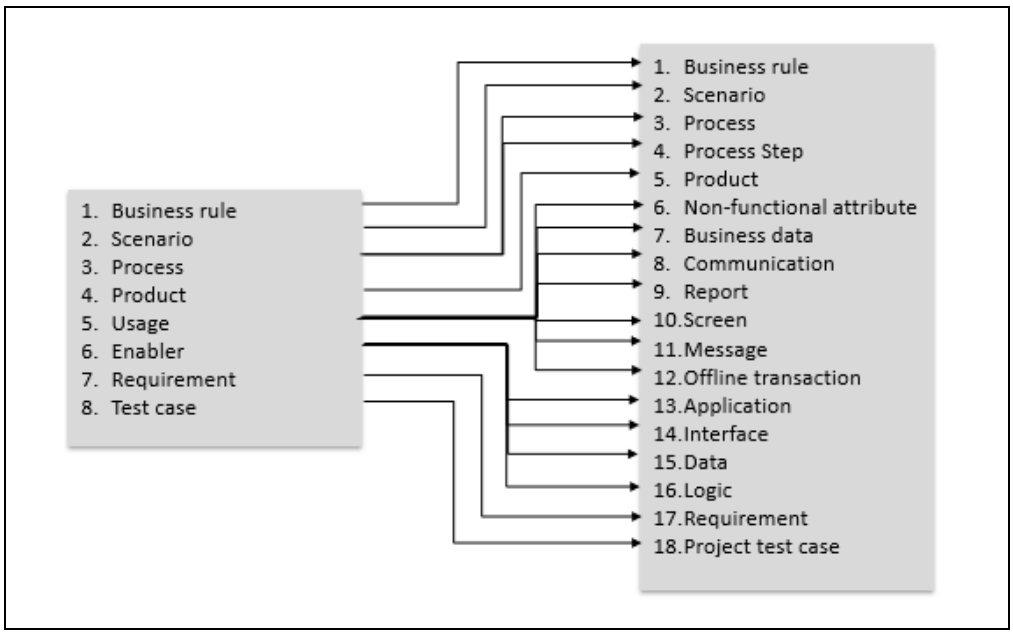

Figure 3: Expanding 8 to 18 building blocks for software development

Usage and Enabler building blocks are expanded into 11 building blocks as listed above to customise it to the software development from enterprise knowledge perspective. This is a typical split and depending on the situation, the total number of building blocks may vary.

For example, for a calculation intensive work, calculation can be a separate building block but in general it is implied in the Rule building block.

These 18 building blocks are reorganised into four constituents of the contextual knowledge relevant to the software development as shown in Figure 4.

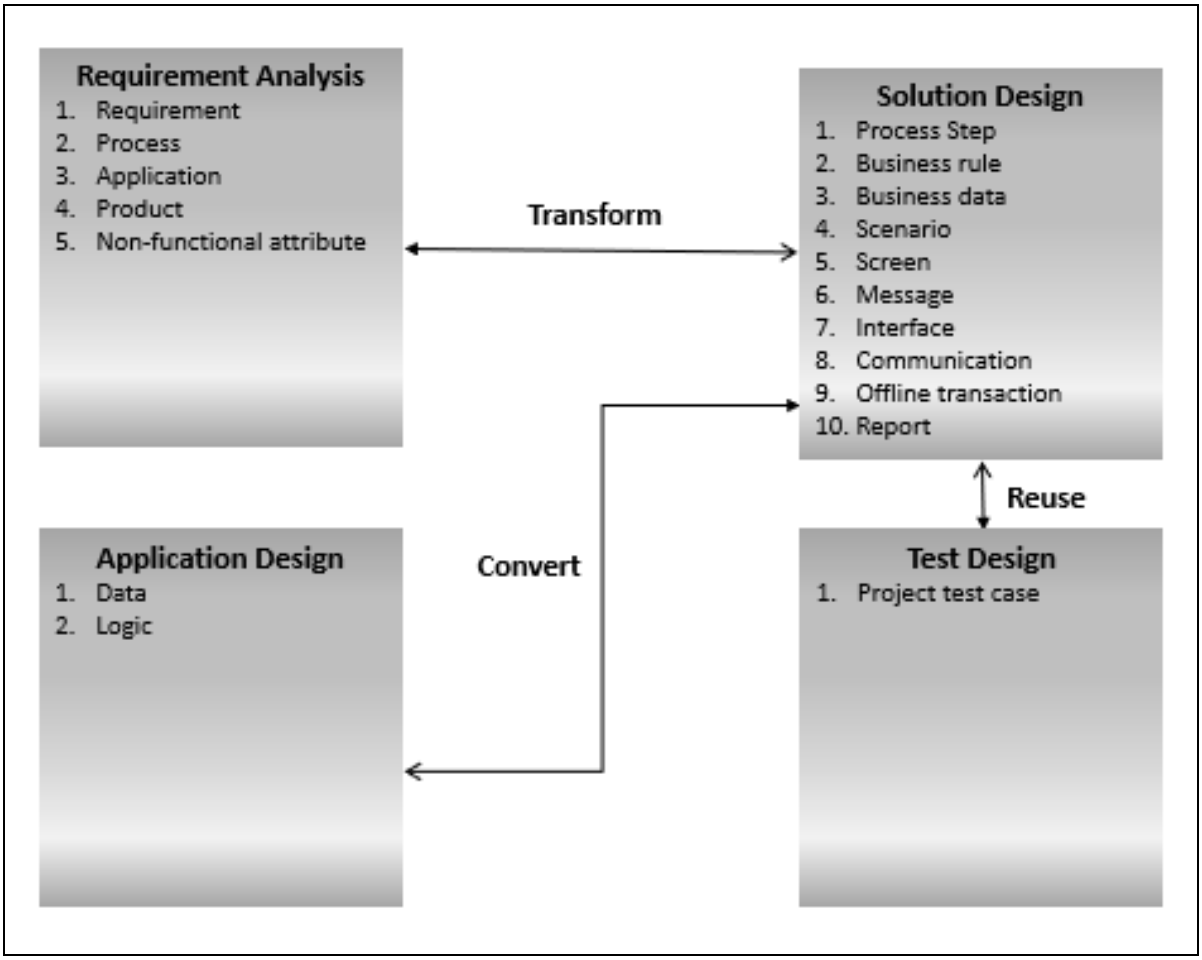

Figure 4: Four constituents of contextual knowledge and its relationship

These four constituents of the contextual knowledge of the software represent complete knowledge from their perspective. If one knows all the 'project test cases', there will be no new knowledge in solution design. These contextual knowledge constituents are also related to each other. Requirements are transformed or expanded into solution design detailing more about the software to be built in business language. Solution design is converted from business language to technical language in application design. Solution design details are reused in test design.

Contextual knowledge specification also assists in execution activities. Execution activities consist of implementing application design (build) and test design (test) knowledge to create a working software. It assists in managing the project knowledge and execution related activities giving rise to a new software development methodology - KDD (Knowledge Driven Development). KDD has a view on important constituents of the software development as listed below:

- Project knowledge – Requirement analysis
- Project knowledge – Solution design
- Project knowledge – Application design
- Project knowledge – Test design
- Project execution – Build
- Project execution – Test execution and defect management
- Project management – Risk management
- Project management – Quality management
- Project management – Task management
- Project management – Configuration management

KDD complies with the current frameworks and standards in software development as listed below:

- Quality assurance framework
- Service management framework
- Testing standards
- Business analysis standards
- Enterprise architecture framework
- Project management framework

Unique selling proposition of KDD

The main proposition of KDD is in digitisation of the project knowledge. Let's understand what I mean by digitisation of the project knowledge. Today, the project knowledge is contained in various sources such as specification documents, presentations, sheets, diagrams, models, wiki. Waterfall methodology places emphasis on exhaustive knowledge capture via specification documents (such as Business Requirement Specification), which is difficult to update regularly. Agile relies mostly on wiki pattern and User Story – Acceptance criteria combination where knowledge remains at high level. I call it analogue project knowledge. KDD aims to digitise this knowledge where the entire knowledge about the software is contained in the inventory and relationship format (detailed later) of 18 building blocks of the project knowledge. IT helps to digitise enterprises and KDD is an attempt to digitise the software development itself.

While Agile and DevOps have brought in much needed speed of delivery in the software development and maintenance, IT industry is still in look out for a better way of managing the contextual knowledge needed for the software development. KDD with its contextual knowledge management proposition fills this much needed gap.

Three important benefits of KDD based on digitisation of the project knowledge are explained below.

1. Digital format resulting in desired quantification in project delivery:

Knowledge of 18 building blocks is specified in the inventory and relationship format as shown in Figure 5.

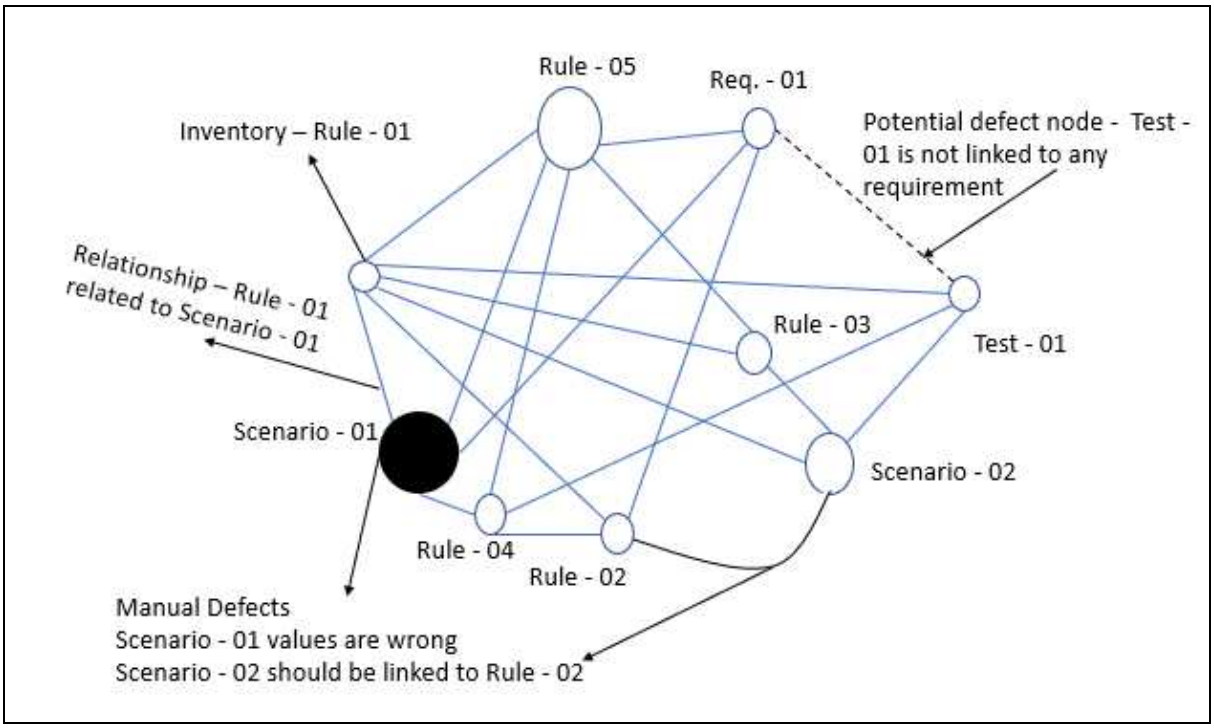

Figure 5: Inventory and relationship format of capturing contextual knowledge

Knowledge is catalogued via inventories of each of the 18 building blocks. For example, requirement-01 and requirement-02 are inventories of requirement building block being specified via attributes such as requirement id, requirement name, requirement description. Linkage between two inventories is known as relationship. An example can be: requirement-01 is related to test-05 (inventory of project test case building block). 18 building blocks give rise to 171 (n*(n + 1)/2) types of

relationship (such as relationship between requirement and project test case, business rule and business data, product and process). Taken together, there are 189 (18 + 171) data points specifying the project knowledge.

This format assists in quality assurance of the project knowledge. Each relationship has two potential nodes of failure. For example, relationship between requirement and project test case building blocks has failure nodes as 'A requirement not linked to any test case' and 'A project test case not linked to any requirement' and this can be derived from the existing relationship. This is a useful information and assists in improving the project knowledge quality. 'A project test case not linked to any requirement' can have either of the two actions – link the project test case to a suitable requirement or delete the test case. Manual review of inventory and relationship assists finding remaining defects in the project knowledge.

Inventory (18), relationship (171) and its quality assurance (171*2=342 potential nodes of failure + 2 (manual review of inventory and relationship) = 344) bring digitisation to the project knowledge. The project knowledge can now be easily measured by a set of activities covering all its digitised constituents making KDD a better choice to manage the project knowledge.

Management thinker Peter Drucker says, 'you cannot manage what you cannot measure'. KDD by providing a mechanism to measure the project knowledge better, aims to solve a well-known problem in the industry on bringing transparency to the project knowledge.

Additionally, the project knowledge management concept of KDD can assist both Waterfall and Agile methodologies.

2. Enabling continuous improvement in project delivery:

Enterprise knowledge is at a higher level of abstraction than the project knowledge. Building blocks such as Product is part of both enterprise knowledge and project knowledge whereas Requirement building block is part of project knowledge only and not the enterprise knowledge.

In KDD, with every project, a portion of project knowledge moves to the enterprise knowledge and this results in the continued growth of the enterprise knowledge. Enterprise knowledge, as it is in the same format (digitised) as project knowledge, can be directly reused in the project delivery environment. The reusability increases with subsequent projects and provides a continuous improvement environment in the project delivery.

3. Supplementing DevOps to cover end to end software development and maintenance:

It is now evident that treatment of knowledge management in KDD is more scientific than Waterfall and Agile methodologies. DevOps is a partnership between software development and IT Operation teams that emphasises communication, collaboration and integration. DevOps takes Agile to the next stage primarily via optimal automation of the execution activities, but it has not given any better treatment to knowledge management than what Agile advocates.

If DevOps uses the knowledge management framework of KDD, end to end software delivery optimisation can truly be achieved as in this case both execution and contextual knowledge activities can be optimised for automation. The digital project knowledge of KDD may also act as an integrator of development and operations team where the project knowledge is created by the development team and maintained by the operations team for the software maintenance. Easier impact analysis for the software maintenance with the help of digital project knowledge will greatly assist the operations team.

Next Steps

GKMF as a new knowledge management framework and KDD as a new software development methodology are explained via 17 chapters of my book at a conceptual level. With more than six months of its publication followed by keen interest generated in academia and in industry, there is a need to take GKMF and KDD to the next stage. The long-term objective of GKMF is to assist in skill development with its digital knowledge management proposition. The long-term objective of KDD is to include it in the relevant courses in the colleges and being used by a portion of the IT workforce. To achieve it, short term next steps can be:

- Do further study to see how GKMF and KDD may assist in Machine Learning, Artificial Intelligence and Data Analytics as I think it may benefit from this new proposition on knowledge management.
- Increase the awareness of GKMF and KDD by publishing articles and white papers in related journals and magazines
- Participating in the leading conferences and seminars on knowledge management, software engineering, digital and Agile and presenting GKMF and KDD
- Create visualisation of GKMF and snippets of its usage in various domains
- Create visualisation of KDD and related case studies
- Organising seminars and workshops on GKMF and KDD

References:

1.Lal, M. (2018). *Knowledge Driven Development: Bridging Waterfall and Agile Methodologies* (Cambridge IISc Series). Cambridge: Cambridge University Press.

2.Lal, M. (2018). *Knowledge Driven Development Supplements DevOps* (Cambridge Core Blog). Cambridge: Cambridge University Press.

About the author

Coming from Darbhanga, Bihar, Manoj Kumar Lal graduated in Mechanical Engineering from BIT Sindri and did his ME in Aerospace from IISc Bangalore in 1997. He joined TCS immediately after IISc and has been there for more than 21 years. A specialist in UK Insurance domain, he has performed almost all the roles in software development. His area of interest is knowledge management and quality assurance.

He is a member of 'International Association for Knowledge Management (IAKM)' and presented in European Conference in Knowledge Management (ECKM), Barcelona in 2017 about a new knowledge management framework. His professional qualifications acquired over the last 20 years include: Certified Scrum Master (CSM) from Scrum Alliance, SAFe 4.0 (Scaled Agile) from Scaled Agile Inc, Foundation Certificate in Software Testing from ISTQB, Foundation Certificate in Business Analysis from British Computer Society (BCS), UK, PRINCE2 foundation certificate from APMG, UK, Certificate in Financial Planning (CFP) and Certificate in Financial Administration (CFA) from Chartered Insurance Institute (CII), UK. He is also an author from IISc Press and Cambridge University Press.

10 Habits of Highly Successful Software Developers

You write clean, reusable code that's easier to read and test
You understand how your code helps drive the overall business
You listen more than you speak—or you at least listen before you speak
You are disciplined
You're able to deeply focus on the right thing
You are a persistent problem-solver
You get help from strangers on the internet
You go beyond skill to achieve expertise, but not necessarily mastery
You are open to new things
You're comfortable with being uncomfortable

Source: https://blog.newrelic.com/culture/successful-software-developers-habits/

12 Most Influential Books Every Software Engineer Needs to Read

Working Effectively with Legacy Code
The Mythical Man-Month
Design Patterns
Programming Pearls (2nd Edition)
CODE: The Hidden Language of Computer Hardware and Software
The Art of Computer Programming
Refactoring
Clean Code
Introduction to Algorithms
Structure and Interpretation of Computer Programs
Pragmatic Programmer
Code Complete

Source: https://jasonroell.com/2015/03/16/12-most-infuential-books-every-software-engineer-needs-to-read/